Microsoft®
Windows™95

CURSO IBM DE PROGRAMACION

15

EN Windows 95

IBM

PLANETA DeAGOSTINI

Microsoft®
Windows™ 95

CURSO IBM DE PROGRAMACION

EN Windows 95

VOLUMEN 2 - UNIDAD 15

Director Gerente: José Mª Puig de la Bellacasa
Director Técnico: Francisco Valer Bañales
Director de Marketing: Javier Graupera Gassol
Director Editorial: Artur Klein Moreno
Directora de Software: Beatriz Cerrato Vidal

Dirección y coordinación: José Luis Becerril Gasco. De la Corporación IBM España, Licenciado en Ciencias Físicas, Director del Instituto de Inteligencia Artificial de IBM España.

Redactores y colaboradores: *José Ramón Dorronsoro Ibero.* Doctor en Ciencias Matemáticas, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid, Investigador del Instituto de Ingeniería del Conocimiento (UAM). *Vicente López Martínez.* Doctor en Ciencias Químicas, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Manuel Alfonseca Moreno.* Doctor Ingeniero de Telecomunicación y Licenciado en Informática, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Javier Garrido Salas.* Doctor en Ciencias Físicas, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Javier Martínez Rodríguez.* Doctor en Ciencias Físicas, Profesor Titular del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Santiago de Aréchaga Tarruell.* Ingeniero Superior de Informática, Investigador del Instituto de Ingeniería del Conocimiento (UAM). *Pedro Pascual Broncano.* Profesor Asociado del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid. *Carlos Aguirre Maeso.* Licenciado en Ciencias Matemáticas, Profesor Asociado del Departamento de Ingeniería Informática de la Universidad Autónoma de Madrid, Investigador del Instituto de Ingeniería del Conocimiento (UAM).

Realización editorial: LH servicios informáticos, S.L.

Diseño gráfico: Esteoeste

© Multimedia Ediciones, S.A., Barcelona
ISBN Obra Completa: 84-88720-79-3
ISBN Fascículos: 84-88720-80-7
ISBN Volumen 2: 84-88720-82-3
Depósito Legal: B-33.798-97

Redacción y administración:
Aribau 198, 5ª planta - 08036 - Barcelona
Tel. (93) 306.78.90 - Fax (93) 306.78.97

Servicio de atención al cliente: Teléfono 902.101.324

Petición de números atrasados:
Teléfono 902.288.288
Apartado de correos nº 2.867,-08080- Barcelona

Servicio de consulta:
Teléfono 902.244.444 de lunes a viernes, de 10.00 a 14.30 y de 15.30 a 19.00 horas.
Fax (91) 661.67.08 durante las 24 horas del día.

Distribuye para España: MIDESA. Marco Ibérica de Ediciones, S.A., Carretera de Irún, km. 13,350, variante de Fuencarral - 28034 - Madrid

Venta en quioscos y librerías: Pida a su proveedor que le reserve sus ejemplares del curso. Adquiriéndolos todas las semanas en el mismo quiosco o librería facilitará la distribución y obtendrá un mejor servicio. El editor se reserva el derecho de modificar el precio de venta de los componentes de la colección en el transcurso de la misma, si las circunstancias del mercado así lo exigieran.

Composición del Curso: El Curso IBM de Programación se publica en forma de 52 unidades de aparición semanal y 52 CD-ROM. Las 52 unidades se encuadernan en 5 volúmenes. Con la unidad que completa cada volumen se ponen a la venta las tapas para su encuadernación.

Multimedia Ediciones garantiza la publicación de todos los elementos que componen esta obra.

© Editorial Planeta Argentina S.A.I.C.
Independencia, 1668 - Capital Federal
Teléfono atención al cliente: 482-0655 y 482-0657
Distribuye en capital: Huesca-Sanabria. En interior: D.G.P.
© Editorial Planeta Mexicana S.A. de C.V.
Avda. Insurgentes # 1162 - México D.F.
Teléfono de atención al cliente: 575-1348
© Editorial Planeta Colombiana S.A.
Carrera 68 A, nº 25-55 - Santafé de Bogotá
Teléfono de atención al cliente: 9800-15315

Imprime: Cayfosa. Carretera de Caldes, km. 3, Santa Perpetua de Mogoda (Barcelona).

INSTALACION/DESINSTALACION DEL SOFTWARE: Para la correcta instalación y desinstalación del CD-ROM que se entrega en esta unidad, véase el libreto que acompaña a cada CD-ROM.

Marcas comerciales registradas: En esta publicación son marcas registradas de sus respectivas compañías: Delphi Desktop, dBase, Turbo Pascal, Turbo C++, Turbo Assembler, Turbo Debugger, Borland Intrabuilder, Paradox, Visual dBase y BRIEF (Borland International, Inc.). Lotus, Lotus 123 y SmartIcons (Lotus Development Corporation, filial de IBM). Microsoft DOS, Microsoft Windows, Microsoft Windows 95, Excel, Visual Basic, Access y FoxPro (Microsoft Corporation). Netscape Navigator Gold (Netscape Communications Corporation). Java (Sun Microsystems, Inc.). IBM (International Business Machines).

CURSO DE PROGRAMACIÓN VISUAL: TURBO PASCAL (2)

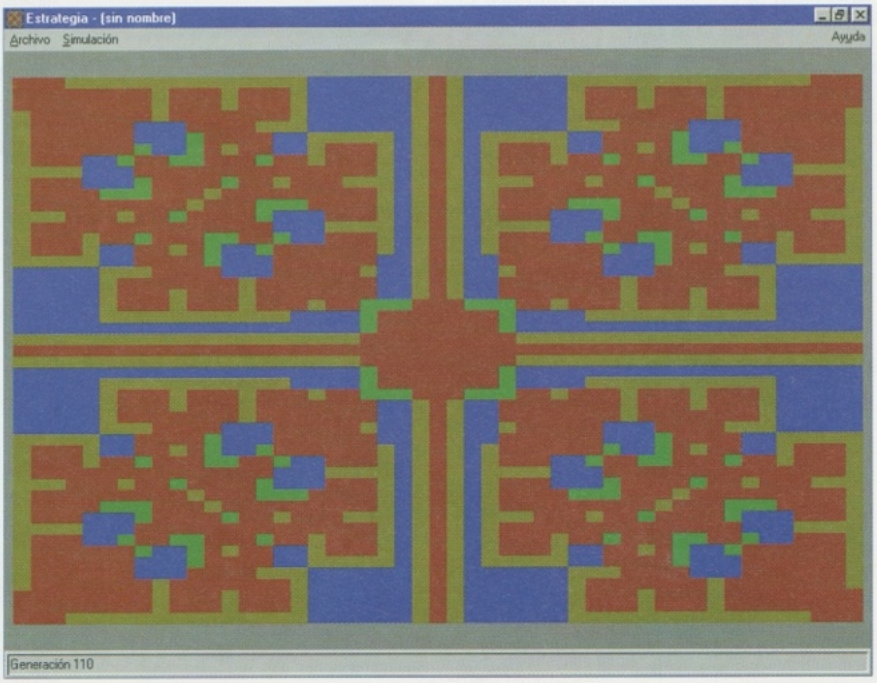
TIPOS DE DATOS Y OPERACIONES EN PASCAL

Realizar operaciones con números es una de las tareas más sencillas que se pueden programar en el ordenador, por ello se utilizarán para estudiar los conceptos básicos de programación relacionados con los tipos de datos de un programa y su manipulación.

El ordenador se programa para procesar información, numérica o no, pero la memoria sólo es capaz de almacenar números como secuencias de ceros y unos agrupados en bytes de información. Además, la Unidad Central de Procesos (UCP) sólo puede realizar operaciones muy básicas con los bytes que se almacenan en la memoria.

Sin embargo, un lenguaje de propósito general de alto nivel, como Pascal, debe poner a disposición del programador un modo de programar la UCP en la que todas esas limitaciones no estén presentes y le permita escribir instrucciones mucho más cercanas al lenguaje natural. Los diseñadores de los lenguajes consiguen llegar hasta ese extremo poniendo a disposición del programador unos tipos de datos básicos para números, letras, y otros caracteres, en los que la forma binaria que adoptan en la memoria del ordenador no se pone de manifiesto. Además, incluyen operadores que permiten programar operaciones básicas, como suma o multiplicación, y con los que el programador se puede olvidar de las particularidades de la UCP.

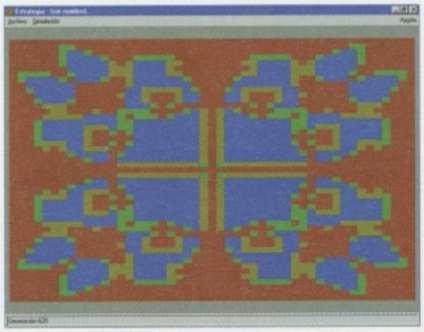
Figs. 1 y 2. Formas geométricas generadas por el programa escrito en lenguaje Pascal, denominado "Estrategia", y que se entrega como ejemplo de programación en Windows.



En esta unidad, en la que se estudiarán los tipos básicos de datos y operaciones, se tendrá que recurrir a veces a detalles muy cercanos a los bytes de la memoria para evitar errores de concepto que puedan dar lugar a fallos en la programación. El lenguaje Pascal resulta especialmente adecuado para ello.

INSTALACIÓN

Este proceso se realiza igual que en las unidades anteriores. Insertar el CD con el programa en el lector del equipo, abrir el menú Inicio y activar la opción Ejecutar. Después, teclear D:\INSTALAR, o dejar que se inicie el programa Autorun automáticamente.



TIPOS BÁSICOS DE DATOS

Para el ordenador, "conocer un dato" significa tenerlo almacenado en la memoria en formato binario, es decir, como secuencia de ceros y unos con la que podrá realizar operaciones. Si bien este modo de "recordar" es uniforme, en un lenguaje de programación resulta muy útil disponer de sistemas para indicar al ordenador que debe recordar y procesar información muy diversa. No sólo debe trabajar con números, ya sean enteros o decimales, sino también con letras, palabras y frases. Para cubrir esas posibilidades, y otras que de-

cida el programador, un lenguaje como el Pascal ha de contar con una clasificación precisa de los tipos de datos. De ese modo, el compilador podrá decidir claramente qué transformaciones son necesarias en cada caso para que toda la información acabe siendo siempre la necesaria secuencia de ceros y unos en la memoria del ordenador.

La variable es el identificador que se utiliza para referirse a una zona de la memoria del ordenador, y en ella se almacena un dato, asignando algún valor a la variable mediante una instrucción de asignación. Pero no todos los datos que hay que recordar en un programa son iguales, por lo que no se pueden guardar en zonas de memoria con el mismo tamaño y, además, no todos se transforman del mismo modo en la secuencia de ceros y unos que reconoce el ordenador. En cada lenguaje de programación existe una sintaxis que permite dejar claro los distintos tipos de datos y cómo se ha de operar con ellos. En Pascal esa sintaxis es muy sencilla y el compilador se encargará de verificar que todo resulte especialmente coherente. Ahora hay que empezar por estudiar los tipos de datos básicos que proporciona por defecto el lenguaje Pascal.

• Datos numéricos

Muchos de los datos que se manipularán en los programas serán números. Entre ellos, hay que hacer una distinción muy clara entre los enteros y los que tengan parte decimal. Para indicar el valor de un número entero en un programa Pascal se hará igual que cuando se escribe en cualquier texto. Por ejemplo, la asignación

```
num := 12;
```

ordena que se guarde el valor doce en la variable "num".

Si no se añade signo, se entiende que se trata de un número positivo; si fuera negativo, habría que incluir delante del valor el símbolo **-**.

Cuando se trate de un número con parte decimal, hay que ponerse de acuerdo sobre el separador de las partes entera y decimal. En Pascal se utiliza un punto.

En este lenguaje los números enteros constituyen el tipo de dato denominado Integer (Entero), mientras que los decimales son del tipo denominado Real. Esta distinción tan marcada entre los dos tipos de datos está impuesta por el hecho de que ambas familias se han de almacenar en la memoria de modo muy distinto.

Cuando en un programa aparece el dato 12, el compilador lo interpreta como dato Integer, mientras que si aparece como 12.0 lo interpretará como Real. No importa que se refieran al mismo número de unidades: el compilador necesita clasificar los datos para poder almacenarlos utilizando la cantidad de memoria adecuada y realizar la transformación óptima a la secuencia de ceros y unos que puede comprender la Unidad Central de Procesos.

El compilador también permite utilizar la notación científica y la letra **E** se utiliza para separar la mantisa del exponente (no hay olvidar que en Pascal es equivalente el empleo de mayúsculas y minúsculas). Por ejemplo, 12.00 también puede indicarse con 0.12E02, 1200.0E-2, o todas las variaciones que se puedan conseguir desplazando potencias de diez del exponente a la mantisa o viceversa. El compilador interpretará que un dato es del tipo Real si encuentra en el número el punto decimal o la E del exponente.

Hay que tener mucho cuidado de no confundir en el teclado el 0 (cero) con la letra o. El compilador no perdonará el error. Tampoco hay que dejar ningún espacio en blanco entre el signo - y el número para indicar que es negativo.

• Caracteres y cadenas de caracteres

Los caracteres que pueden comunicarse al ordenador utilizando el teclado constituyen otro tipo de dato básico del Pascal que se denomina Char (por las primeras letras de la palabra inglesa Character, carácter). Cuando en un programa se quiera utilizar uno de estos caracteres como dato del tipo Char, se colocará entre comillas simples('). Por ejemplo, la expresión 'a' se refiere a la primera letra del alfabeto en minúscula.

Todos los símbolos del teclado también se almacenan en la memoria del ordenador como una secuencia de ceros y unos. Para cada símbolo hay asociado un número que es el especificado por la tabla de correspondencias ASCII. Por tanto, está claro que es muy distinto el dato 5 (Integer) que el dato '5' (Char), cuando aparecen en un programa Pascal.

El dato Integer se utilizará para hacer referencia al número 5 en operaciones aritméticas, y el dato Char para referirse al carácter tal como se puede necesitar para formar un texto. El primero se guardará en la memoria con la forma binaria del número 5, y el segundo con la forma binaria del 97, que es el número de orden que le corresponde en la tabla ASCII.

Las letras casi siempre se utilizan formando parte de palabras. En el tipo de dato String (su traducción en español es cadena, por cadena de caracteres) se pueden almacenar varios de ellos a la vez de una forma ordenada. Se utilizan también las comillas simples para delimitar la extensión de un dato String, y formarán parte de él todos los caracteres que se encuentren entre la primera y última comilla que lo delimitan.

Por ejemplo, con la asignación

```
lugar := 'La noria';
```

se está guardando en la variable lugar una cadena con 8 caracteres, puesto que el blanco de separación es otro carácter más (el que hace el número 32 en la tabla ASCII).

Si se analiza, podrá verse que también es posible la existencia de dos símbolos ' seguidos; esto sirve para indicar que la comilla forma parte integrante de la cadena de caracteres y no es el delimitador de ese tipo de dato. Por ejemplo, la salida que produce

```
WriteLn('Erik''s mother')
es:
Erik's mother
```

Se utlizará ahora el programa de la figura 4 para ilustrar los diferentes modos de imprimir una frase utilizando datos de tipos distintos.

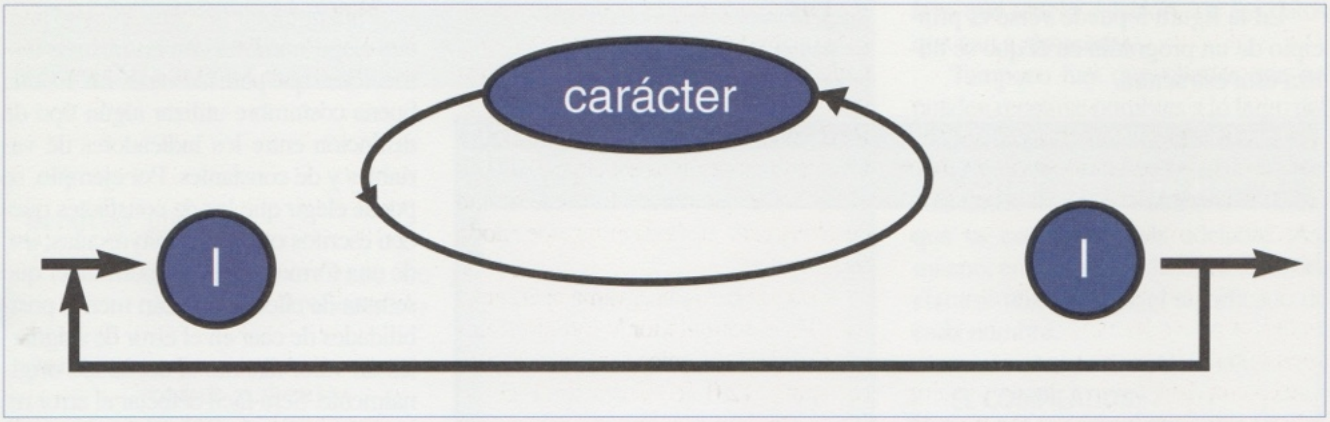


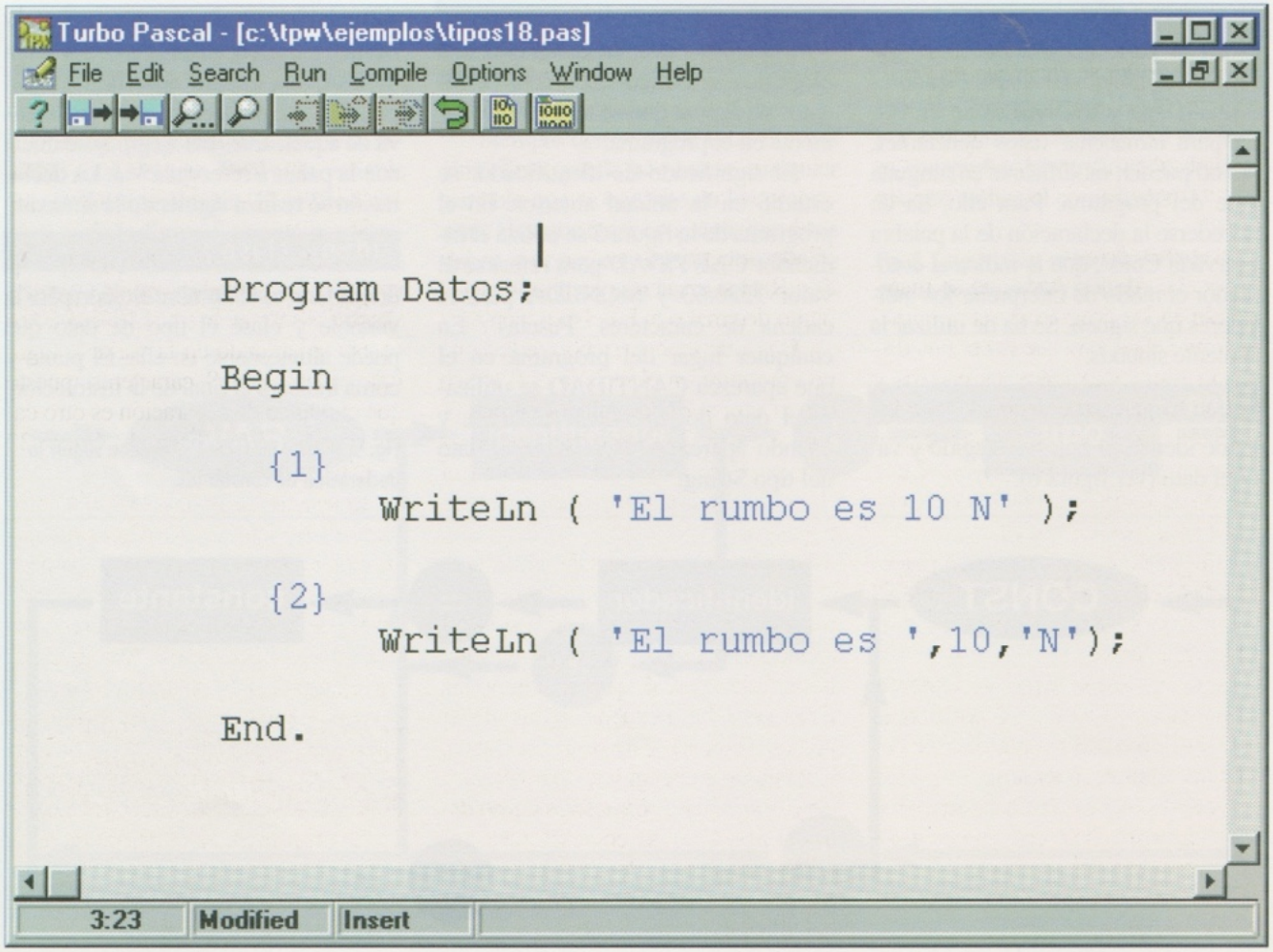
Fig. 3. Gráfico sintáctico que define un dato del tipo String en Pascal.

En la primera llamada a la función WriteLn sólamente se incluye un argumento: un dato del tipo String. Sin embargo, en la siguiente instrucción se utilizan tres argumentos: el primero del tipo String, el segundo del tipo Integer, y el último del tipo Char.

• **Constantes**

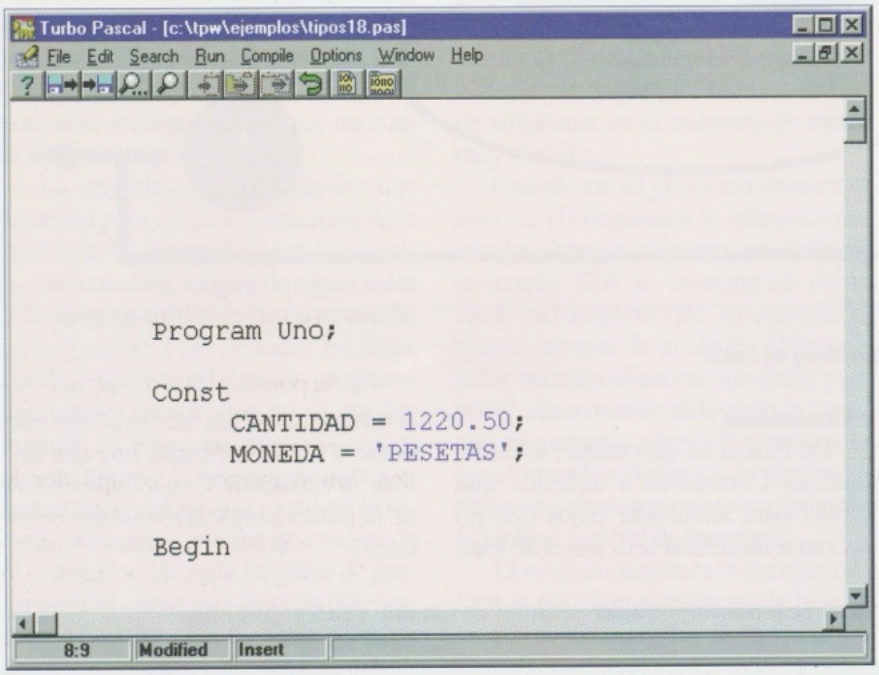
En Pascal se denominan identificadores Constantes a aquéllos que sirven para almacenar datos que no se van a modificar a lo largo de todo el programa. Para declarar de este modo a un identificador hay que utilizar la estructura Const (abreviatura de la palabra inglesa Constant, constante).

Fig. 4. En la primera instrucción se escribe un solo dato y en la segunda tres.



En la figura 5 puede verse el principio de un programa en el que se utiliza esta estructura.

Fig. 5. Ejemplo de la declaración de constantes al principio de un programa.



En la parte declarativa, antes del Begin, se indica que los indicadores CANTIDAD y MONEDA se utilizarán para almacenar datos constantes, que no pueden modificarse en ninguna parte del programa. Para ello, ha de precederse la declaración de la palabra reservada Const, que le indica al compilador el modo de interpretar los indicadores que siguen. Se ha de utilizar la siguiente sintaxis:

```
iden = valor ;
```

donde iden es el nombre elegido y valor el dato (ver figura 6).

Por constante se ha de entender en el gráfico sintáctico cualquiera de los datos de Pascal que se tecleen directamente en el programa.

El significado de identificador se estudió en la unidad anterior. En el programa de la figura 5 se utiliza el indicador CANTIDAD para referirse al valor 1220.50 y MONEDA para la cadena de caracteres 'Pesetas'. En cualquier lugar del programa en el que aparezca CANTIDAD se utilizará el dato del tipo Real 1220.50, y cuando aparezca MONEDA, el dato del tipo String.

Para nombrar las constantes se pueden elegir nombres con las mismas restricciones que para las variables. Es una buena costumbre utilizar algún tipo de distinción entre los indicadores de variables y de constantes. Por ejemplo, se puede elegir que los de constantes queden escritos con letras mayúsculas; así, de una forma rápida se podrá saber que se trata de ellos y existirán menos posibilidades de caer en el error de asignarles un valor distinto al declarado originalmente. Será fácil detectar el error revisando que no haya un indicador escrito con mayúsculas en la parte izquierda de una declaración situada en el cuerpo del programa (después del Begin).

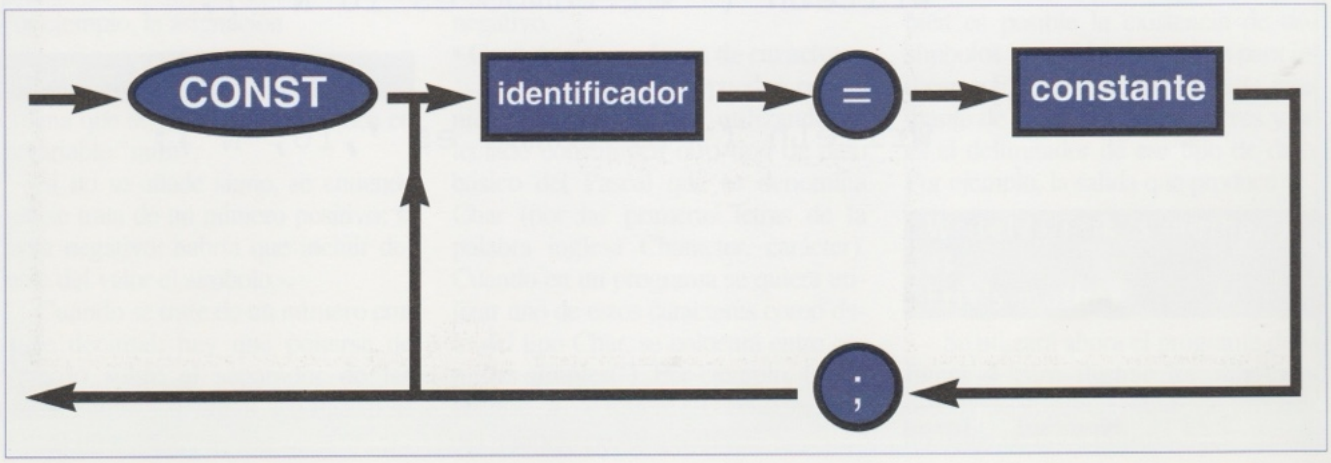
- **Variables**

También se han de incluir en la parte declarativa del programa las declaraciones de los nombres con los que se hace referencia a los datos variables. Es decir, los indicadores que se refieren a los lugares de la memoria del ordenador en los que se indicará al programa que almacene datos. La declaración de las que van a ser las variables del programa se sitúa en la parte declarativa de aquél, antes del Begin, y se inicia con la palabra reservada Var. La declaración se realiza siguiendo la sintaxis:

```
iden : clase ;
```

donde iden es el nombre elegido para la variable y clase el tipo de dato que puede almacenarse en ella. El punto y coma indicará el final de la instrucción.

Fig. 6. Gráfico sintáctico que debe seguir la declaración de constantes.



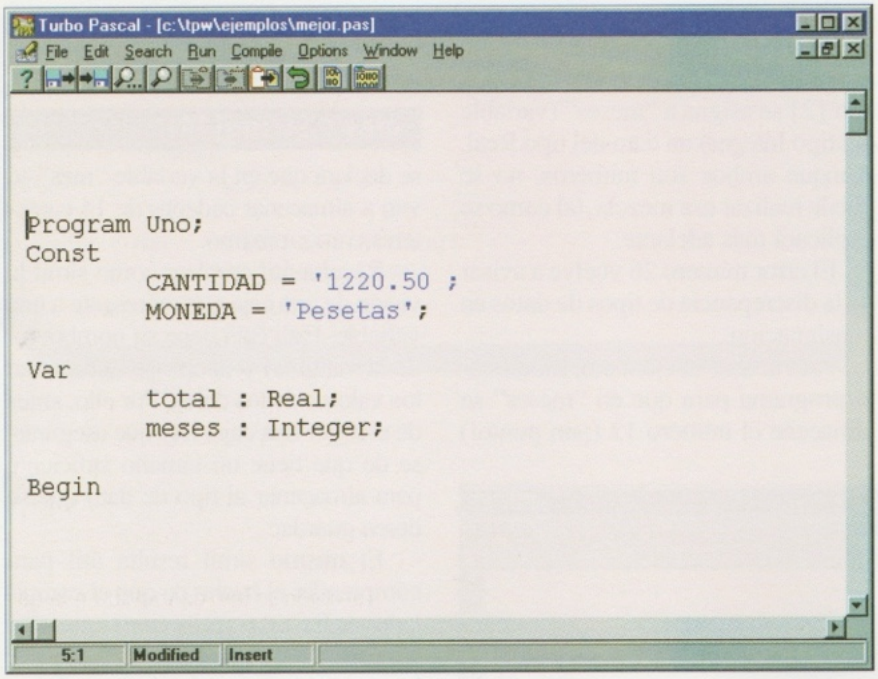


Fig. 7. Al principio de este programa se declaran dos indicadores de constantes y dos de variables.

Para ilustrar el uso de estas declaraciones, se va a ampliar el principio del programa Uno hasta el que puede ver en la figura 7.

En ese programa se declara la variable total del tipo Real y la variable meses del tipo Integer. El gráfico sintáctico de la figura 8 servirá para analizar posibles declaraciones válidas de identificadores variables.

Por "tipo" se entiende en el diagrama cualquiera de los tipos de datos descritos anteriormente (Char, Integer, Real,...).

Siempre que se sigan estas normas sintácticas, puede nombrarse cualquier variable que se necesite en los programas. Hay que tener en cuenta que éstos deben ser comprensibles y, para ello, lo mejor es utilizar para las variables nombres que tengan relación con los datos que van a almacenar.

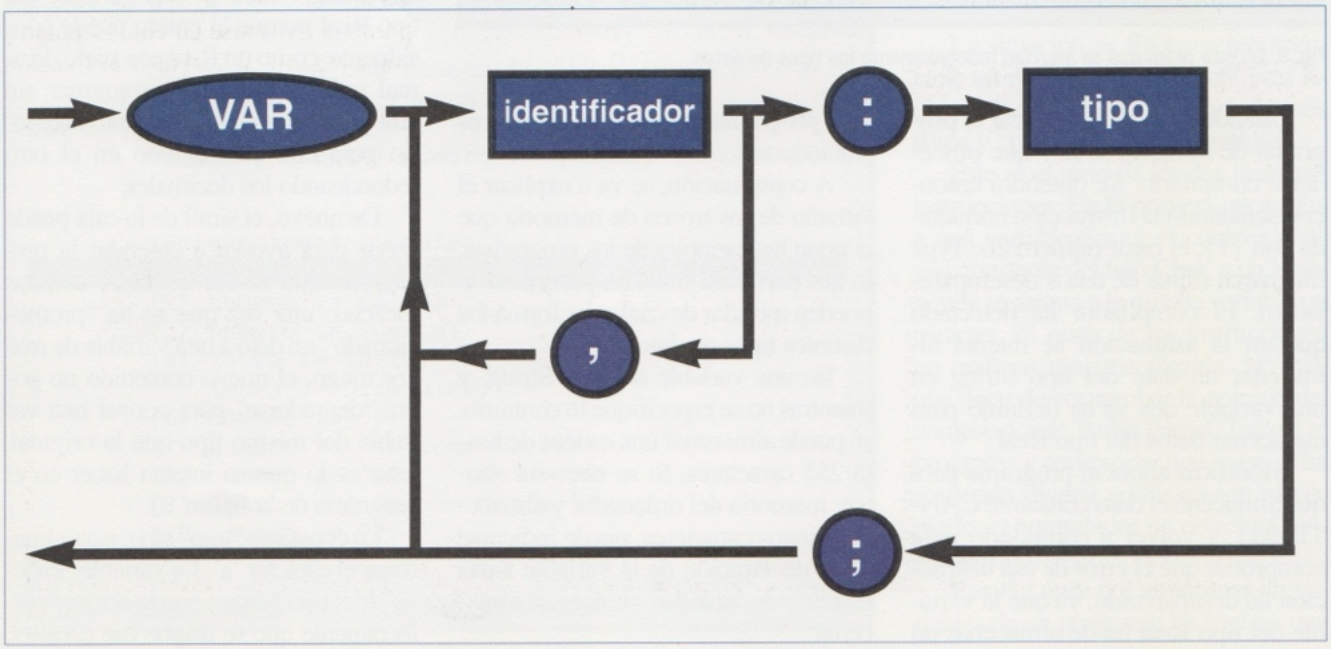
Tampoco hay que olvidar que no pueden repetirse nombres a lo largo del programa. A cada nombre ha de corresponder un solo indicador, ya sea constante, variable, o cualquiera de los que se estudiarán más adelante. Así mismo, en la parte declarativa debe declararse una sola vez el significado de cada nombre.

EL COMPILADOR: UN COMPAÑERO DE FATIGAS

Es responsabilidad del programador hacer un uso coherente a lo largo del programa de las variables declaradas. Sin embargo, el compilador le ayudará a repasar aquél, señalándole algunas de las incoherencias. A continuación, se va a explicar hasta qué punto puede ayudar. Para ello, hay que empezar una sesión de Turbo Pascal y teclear el programa que aparece en la figura 9.

El programa bautizado con el nombre de "Francamente_Mal" es inútil por varios motivos, pero va a servir para comprobar la verificación de sintaxis que realiza el compilador.

Fig. 8. Las variables se han de declarar siguiendo este gráfico sintáctico.



TRUCO

Si se quiere cambiar en Turbo Pascal el tipo letra con el que aparecen escritos los programas en la ventana de edición, elegir la opción Options|Preferences... (Opciones|Preferencias). Modificar la selección del tipo de letra (Font). La que se ha utilizado en la figura 9 es Courier New del tamaño 16.

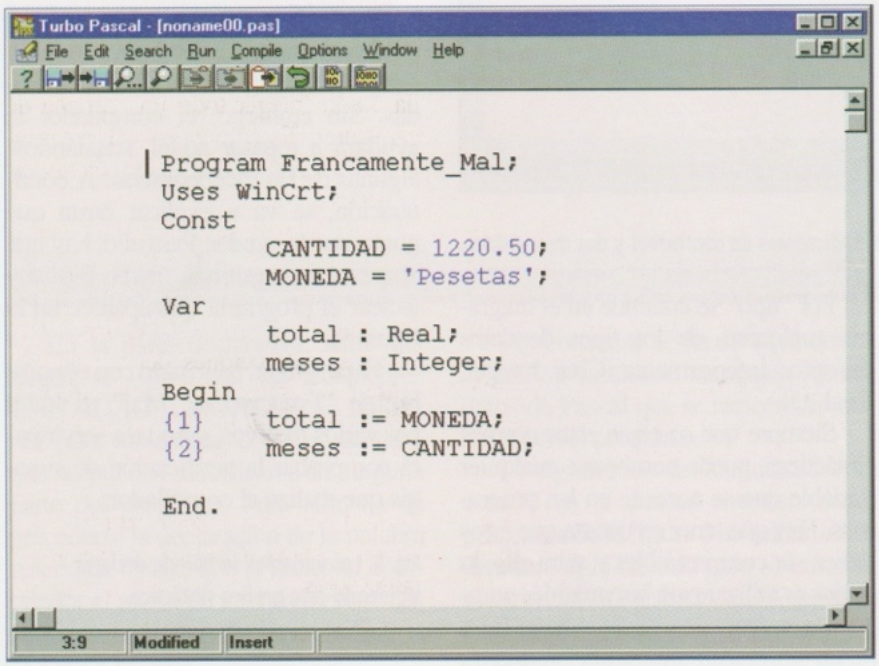


Fig. 9. En este programa se mezclan indebidamente los tipos de datos.

Cuando se acabe de teclear el programa de la figura 9, hay que proceder a compilarlo. Se obtendrá entonces, señalando la instrucción etiquetada con {1}, el error número 26: Type mismatch (tipos de datos desemparejados). El compilador ha detectado que en la asignación se intenta almacenar un dato del tipo String en una variable que se ha definido para almacenar datos del tipo Real.

Modificar ahora el programa para que almacene el dato constante CANTIDAD, y volver a compilarlo para comprobar que el error de esa instrucción ha desaparecido, ya que la variable del tipo Real ha de almacenar un dato (constante) del mismo tipo. Sin embargo, el compilador detecta ahora otro error: en la instrucción etiquetada con {2} se asigna a "meses" (variable del tipo Integer) un dato del tipo Real. Aunque ambos son números, no se puede realizar esa mezcla, tal como se explicará más adelante.

El error número 26 vuelve a avisar de la discrepancia de tipos de datos en la asignación.

Para acabar el ejercicio, modificar el programa para que en "meses" se almacene el número 12 (¡sin punto!) y el programa estará libre de errores sintácticos.

A continuación, se va a explicar el tamaño de los trozos de memoria que ocupan las variables de los programas, lo que permitirá entender porqué no se pueden mezclar de cualquier forma los distintos tipos de datos.

En una variable del tipo String, y mientras no se especifique lo contrario, se puede almacenar una cadena de hasta 255 caracteres. Si se necesita ahorrar memoria del ordenador y almacenar menos caracteres, puede indicarse en la declaración de la variable hasta cuántos de aquéllos se desean almacenar.

El número se ha de situar entre corchetes y a continuación de la palabra reservada String. Con

```
mes : String[14];
```

se declara que en la variable "mes" se van a almacenar cadenas de 14 caracteres como máximo.

Resulta útil emplear como símil la visión de una caja que represente a una variable. Esta caja tiene su nombre (el de la variable) y dentro se almacenan los valores de los datos. Por ello, antes de escoger una caja, hay que asegurarse de que tiene un tamaño suficiente para almacenar el tipo de dato que se desea guardar.

El mismo símil resulta útil para comprender el hecho de que el compilador de Pascal permite ciertas mezclas de tipos de datos en las asignaciones. La condición que se ha de cumplir para que esté permitida la mezcla depende del tamaños de los datos y los lugares de memoria reservados para las variables.

Esto se puede expresar coloquialmente del siguiente modo: en una variable se puede almacenar un dato de un tipo distinto al declarado sólo si éste cabe en la caja. Por ejemplo, un dato del tipo Char cabe en una variable del tipo String, ya que ésta admite hasta 255 datos del tipo Char. Un dato del tipo Integer cabe en una variable del tipo Real porque el entero puede considerarse como un Real con parte decimal nula, y se puede programar sin ambigüedades el truncamiento necesario para convertir el uno en el otro redondeando los decimales.

De nuevo, el símil de la caja puede servir para ayudar a entender la precaución que se ha de tener con las mezclas: una vez que se ha "promocionado" un dato a una variable de mayor rango, el nuevo contenido no podrá "degradarse" para ocupar una variable del mismo tipo que la original. Esto es lo que se intenta hacer en el programa de la figura 10.

La constante "uno" sirve para almacenar el carácter 'a'. La variable "cadena" (del tipo String) puede aceptar perfectamente que se asigne ese carácter,

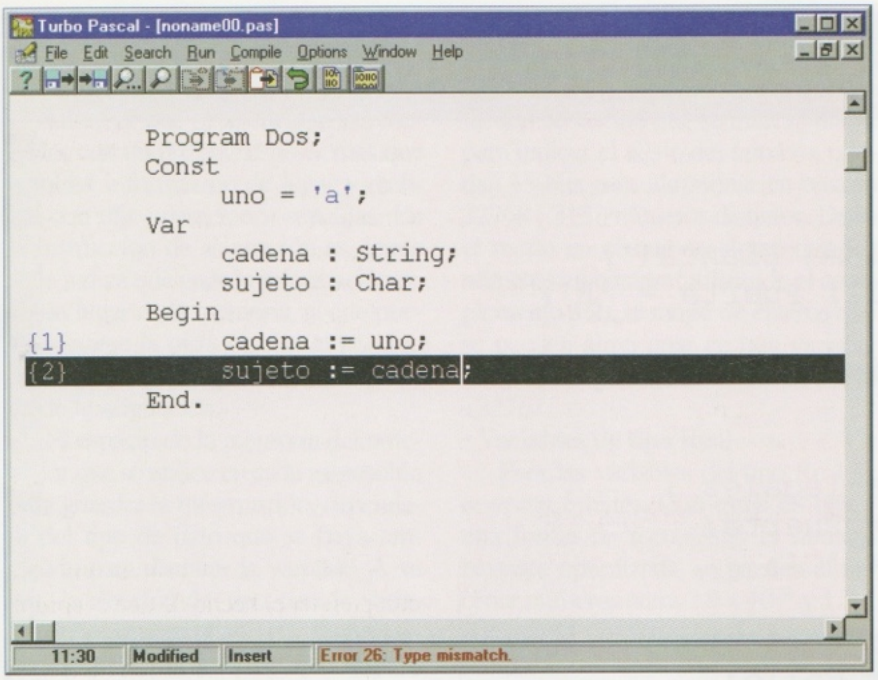


Fig. 10. Mezcla inoportuna de datos.

pero la variable "sujeto" (del tipo Char) será incapaz de almacenar el contenido de "cadena". No importa que la información almacenada en "cadena" sea solamente (en apariencia) la 'a' original de la constante "uno".

En un principio la variable "cadena" podría contener hasta 255 caracteres distintos, pero no existe una forma sencilla de decidir cuál es la que hay que retener cuando se han de condensar todas ellas en un solo carácter. El ordenador no puede adivinar los deseos del programador.

El compilador detectará la mezcla que no está permitida y avisará con el error número 26 de la discrepancia de tipos de datos (figura 8).

A continuación, se va a analizar un programa en el que se incluyen asignaciones con mezclas de tipos de datos. Se trata del programa "Fatal" (ver figura 11).

Todas las instrucciones de este programa son incorrectas y el compilador empezará protestando en la primera, que ya analiza con el error número 26. En la primera instrucción se trata de almacenar un dato Integer en una variable del tipo String. En la segunda se sigue con la racha de errores, queriendo almacenar en una variable declarada como Integer todo una cadena de caracteres. Es conveniente ir repasando los errores presentes en cada una de las instrucciones.

Para ello, se puede obtener ayuda de un programa con la misma parte declarativa que el presentado en la figura 11, al que pueden ir añadiéndose una a una las instrucciones del cuerpo del programa "Fatal". En todas ellas el compilador tendrá que avisar del error en la mezcla de datos.

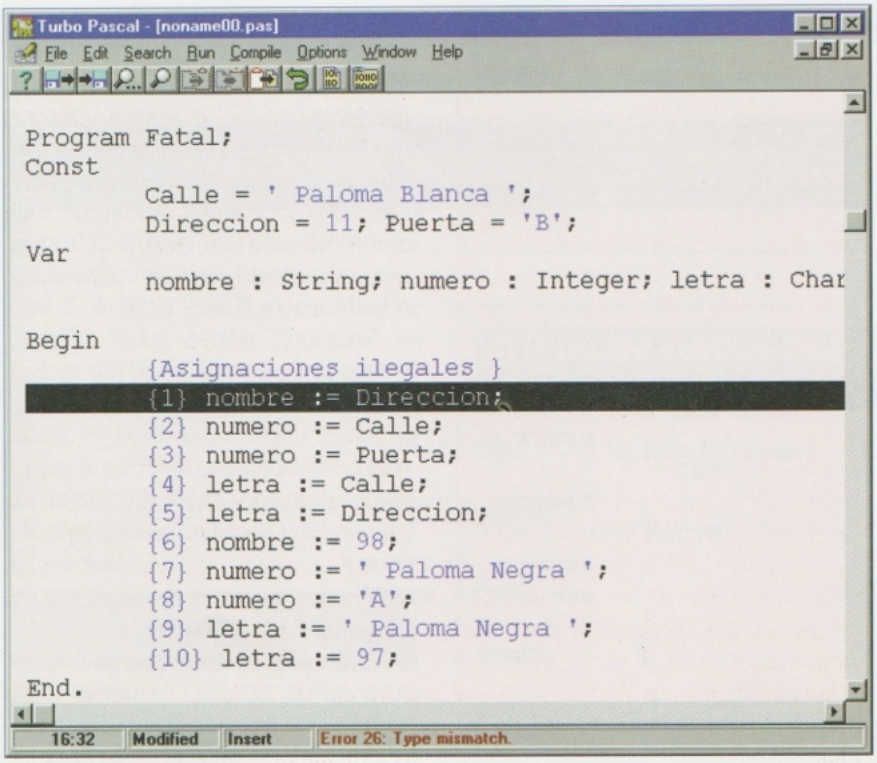


Fig. 11. Programa en el que se mezclan erróneamente tipos de datos en todas las instrucciones.

La situación es francamente aceptable en el programa "Mejor" (ver figura 12), donde se mezclan algunos datos y "promociones" permitidas.

Éste es el caso de las dos primeras instrucciones. En la primera, se asigna a una variable String el contenido de una constante del tipo Char, y en la segunda se asigna a la misma variable un carácter. El resto de las instrucciones no realizan ninguna mezcla. No hay que dejar de comprobar la opinión del compilador de Turbo Pascal. Teclear el programa y comprobar que, si no se ha cometido alguna equivocación al copiarlo, el compilador no detectará ningún error.

Resulta muy útil almacenar en variables de un programa las lógicas que

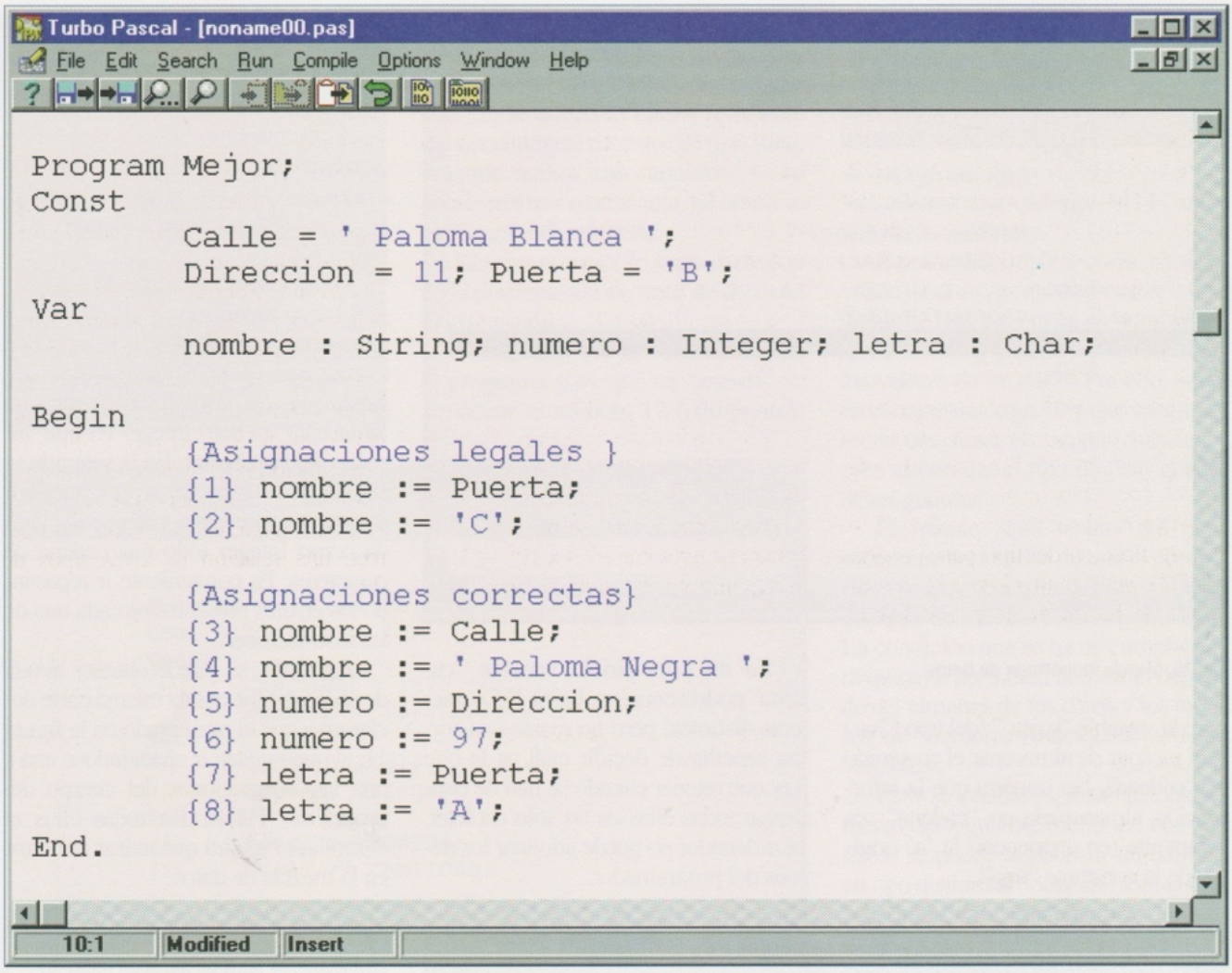
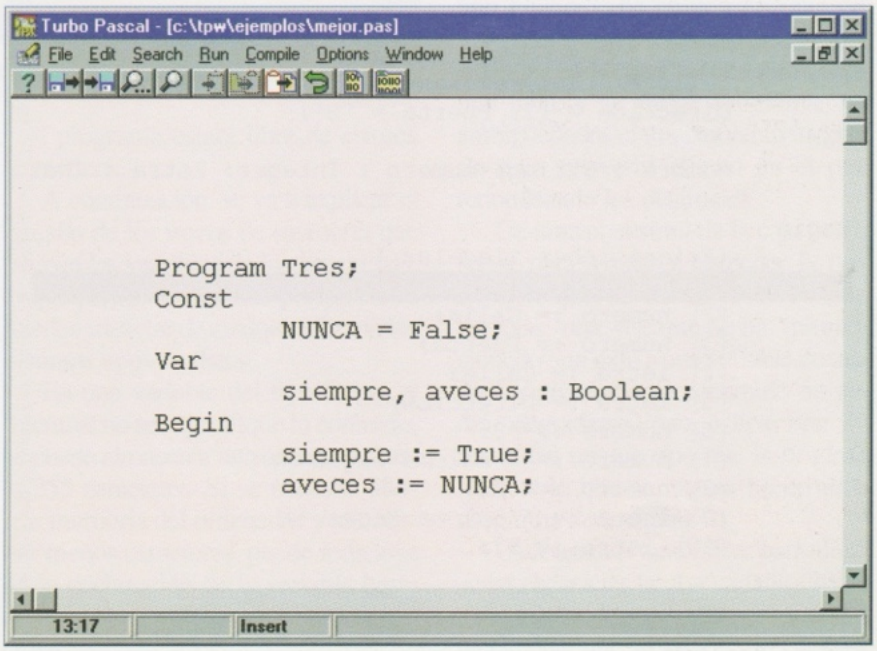


Fig. 12. El compilador no detectará errores en ninguna de las asignaciones de este programa.

indican si una afirmación es cierta o falsa. Para ello, Pascal cuenta con las variables denominadas Boolean, en honor del eminente matemático Gorge Boole. Los posibles valores que puede tomar una variable de este tipo son "Cierto" o "Falso" y se utilizan muy a menudo para elegir entre posibles opciones. Los dos valores que pueden tomar las variables Boolean son los de las constantes que se expresan mediante **True** (Cierto) y **False** (Falso), siguiendo la nomenclatura inglesa. En el principio del programa que aparece en la figura 13 se asigna a "siempre" el valor True (cierto), y a la variable "a veces" el valor almacenado en "NUNCA", que se ha declarado que sea siempre el valor False (Falso).

Fig. 13. Ejemplo de utilización de variables y constantes del tipo Boolean.



VARIABLES: RANGO Y TAMAÑO DE MEMORIA

La labor que se le encarga al ordenador con un programa no es más que cambiar información de lugar y realizar con ella operaciones sencillas. En la instrucción de asignación es donde se le indica que guarde información en algún lugar de la memoria, al que puede seguirse la pista gracias al nombre de la variable que aparece a la izquierda de la asignación.

El espacio de la memoria del ordenador que se utilice en cada asignación para guardar la información, dependerá del tipo de dato que se haya empleado para declarar la variable. A su vez, la elección del tipo para la variable debe venir marcado por las necesidades de la información que vaya a contener.

No hay que olvidar que el tamaño no es flexible en los tipos de datos básicos del Pascal y, por tanto, para evitar errores es muy conveniente conocer tamaño y capacidad de almacenamiento de los distintos tipos de variables que se pueden declarar.

• Variables de caracteres

Un caso muy ilustrativo es el de las variables del tipo Char. Para ellas se reserva un byte de memoria. Si cada byte contiene 8 bits, en un byte se pueden almacenar 256 (28) números distintos en binario. En el caso de los datos del tipo Char se utilizan los 8 bits para almacenar números desde el 0 hasta el 255. A su vez, esos 256 números se utilizarán para identificar caracteres de la tabla ASCII extendida. Por ejemplo, si la variable "número" se declara del tipo Char y se le asigna el valor '7' (sin olvidar las comillas sencillas), el byte de memoria al que se refiere la variable contendrá la secuencia de bits 00110111, que corresponde a la representación binaria del número 55, ya que

$$55 = 1 \times 25 + 1 \times 24 + 0 \times 23 + 1 \times 22 + 1 \times 21 + 1 \times 20$$

El número 55 aparece porque es el número de orden que ocupa el carácter '7' en la tabla ASCII.

• Variables de tipo Entero

Para las variables Integer el compilador reserva 2 bytes de memoria. Teniendo en cuenta que se reserva un bit para indicar el signo del número, quedan 15 bits para almacenar en binario 32768 (215) números distintos. Dado el modo en el que se almacenan los números en binario (utilizando el complemento a 2), el rango de enteros que se pueden almacenar en una variable del tipo Integer va desde -32768 hasta +32767.

• Variables de tipo Real

Para las variables del tipo Real se reservan 6 bytes. Con estos 48 bits y una forma de representar el número bastante optimizada, se pueden almacenar números entre 2.9×10^{-39} y 1.7×10^{38}, tanto positivos como negativos. Hay que resaltar que debido a la representación interna utilizada, se almacenan con igual precisión (11 ó 12 cifras significativas) todos los números en ese rango.

• Otros tipos de datos

Pero aquí no acaban todas las opciones que ofrece Turbo Pascal como tipos de datos predefinidos. Para que se puedan declarar variables adecuadas a rangos de números muy distintos, el programa dispone de diferentes variaciones de los tipos de datos **Integer** y **Real** con mayor o menor espacio de memoria asignada, distinta representación interna de los números y, por tanto, con distinta capacidad de almacenamiento. En los recuadros aparece una relación de estos tipos de datos.

TIPOS DE DATOS PARA LOS NÚMEROS ENTEROS

Tipo	Tamaño (en Bytes)	Rango	Signo
Byte	1	0 .. 255	no
ShortInt	1	-128 .. 127	sí
Word	2	0 .. 65535	no
Integer	2	-32768 .. 32767	sí
LongInt	4	-2147483648 .. 2147483647	sí
Comp	8	$-2^{63} + 1$ .. $2^{63} - 1$	sí

TIPOS DE DATOS PARA LOS NÚMEROS CON DECIMALES

Tipo	Tamaño (en Bytes)	Rango	Cif. signif.
Single	4	1.5×10^{-45} .. 3.4×10^{38}	7 - 8
Real	6	2.9×10^{-39} .. 1.7×10^{38}	11 - 12
Double	8	5.0×10^{-324} .. 1.7×10^{308}	15 - 16
Extended	10	3.4×10^{-4932} .. 1.1×10^{4932}	19 - 20

Se puede pensar que el buen estilo de programación implica utilizar siempre las variables de un tipo muy ajustado en capacidad a los valores que se espera que contengan. Pero no es así. El extremar la variedad de tipos de variables puede dar lugar a errores en la programación de operaciones con mezclas de datos y obliga a estar muy atento a las declaraciones de las variables.

En general, resulta más adecuado utilizar un solo tipo para los enteros y uno solo para los números con decimales a lo largo de un mismo programa. únicamente conviene ajustar los tamaños de las variables cuando se vayan a utilizar muchas iguales, pues una declaración demasiado amplia de estos tamaños puede dar lugar a problemas porque se exceda la capacidad de memoria del ordenador.

• **Datos lógicos**

Los datos Boolean ocupan un byte de memoria. Aunque sea un desperdicio, ya que un bit es suficiente para decidir entre cierto o falso, la arquitectura del ordenador personal hace más eficiente el almacenamiento en múltiplos de bytes.

Con esto, ya se han explicado los tipos de datos básicos que existen en el lenguaje Pascal. Con ellos pueden especificarse en los programas los valores con los que se habrán de realizar operaciones. También se han estudiado las variables que se utilizan para almacenar resultados intermedios de estas operaciones y, en la unidad anterior, los rudimentos de la comunicación entre el programa y el usuario. A continuación, se van a explicar todas las operaciones que se pueden realizar con los datos en un programa de Pascal.

OPERADORES ARITMÉTICOS Y EXPRESIONES ARITMÉTICAS

Cuando los datos con los que se realizan operaciones son números, lo más común es que haya que efectuar operaciones aritméticas: suma, resta, multiplicación y división.

En Pascal, estas operaciones se especifican con los operadores + , - , * , y / , respectivamente. En la instrucción:

```
x := 27.0 + 35.3 ;
```

se realiza la suma de los datos del tipo Real que se especifican directamente en el programa y el resultado se almacena en la variable que aparece en la parte izquierda de la asignación y que se habrá declarado previamente que sea del tipo Real.

En la instrucción:

```
x := z * 3.3 ;
```

se asigna a la variable "x" el resultado de multiplicar el contenido de la variable "z" y el dato 3.3, del tipo Real. Ambas variables "x" y "z" se habrán tenido que declarar del tipo Real.

En la instrucción:

```
x := -x / z ;
```

se asigna a la variable "x" el resultado de dividir el contenido de la variable "x" hasta ese momento (con el signo cambiado) y el contenido de la variable "z".

Todos estos son casos sencillos en los que solamente dos operandos se asocian con un operador aritmético, sin dar lugar a ningún tipo de ambigüedad. Sin embargo, cuando se desea operar con más de dos datos hay que especificar claramente el orden en el que se quieren realizar las operaciones. Para ello, el usuario puede ayudarse de los paréntesis, utilizándolos igual que en las expresiones aritméticas. Por ejemplo, la siguiente instrucción no es ambigua:

```
z := ( -7.2 * (x - 332.0) )
/ (x + 98.);
```

gracias al uso generoso de los paréntesis. Si se eliminan todos ellos, se obtiene la instrucción

```
z := -7.2 * x - 332.0 / x + 98;
```

que el compilador interpretará siguiendo las prioridades establecidas por defecto en el Pascal. El orden viene determinado por los operadores que aparecen en las expresiones y sobre los operandos se aplican con esta prioridad:

1.- más y menos monarios.
2.- paréntesis.
3.- multiplicación y división.
4.- suma y resta.

El "Más y el menos monarios" son los operadores indicadores del signo de los números (positivos o negativos) y tienen prioridad sobre cualquier otro. Las siguientes instrucciones son equivalentes:

```
b:= -27.0 * c + 35. / a + b + c;
b:= ( (-27.0) * c ) + ( 35. / a ) + b + c;
b:= ( -27.0 * c ) + ( 35. / a ) + b + c;
```

y la forma concreta en la que el programador debe escribir una expresión es aquélla que le resulte menos confusa. Conviene recordar que el compilador no se confunde y sigue siempre sus reglas de operación y que, además, el uso generoso de los paréntesis ahorrar el tiempo que habría que gastar después arreglando programas.

Un error común entre principiantes es escribir la instrucción

```
x := a * 2. / b + c ;
```

en lugar de la correcta:

```
x := a * 2. / ( b + c ) ;
```

Una vez aprendida la utilización de los operadores aritméticos ya se pueden empezar a construir programas con cierta utilidad. Al menos, será posible escribir los que realicen cálculos sencillos, los utilizados con mayor frecuencia. Como ejercico práctico, escribir un programa que realice el cálculo del Impuesto del Valor Añadido (IVA). Se trata sencillamente de incrementar en un porcentaje del 16% la cantidad que se grava con el impuesto. El programa podría ser el que aparece en la figura 14.

Al analizarlo podrá comprobarse que se entienden casi todas sus instrucciones: las declaraciones de constantes y variables, la lectura de datos y las operaciones. Sólo falta explicar un detalle nuevo que aparece en la escritura de resultados. En los argumentos de la función "WriteLn", después de los nombres de las variables cuyo contenido se ha de hacer constar, se indica cómo han de escribirse

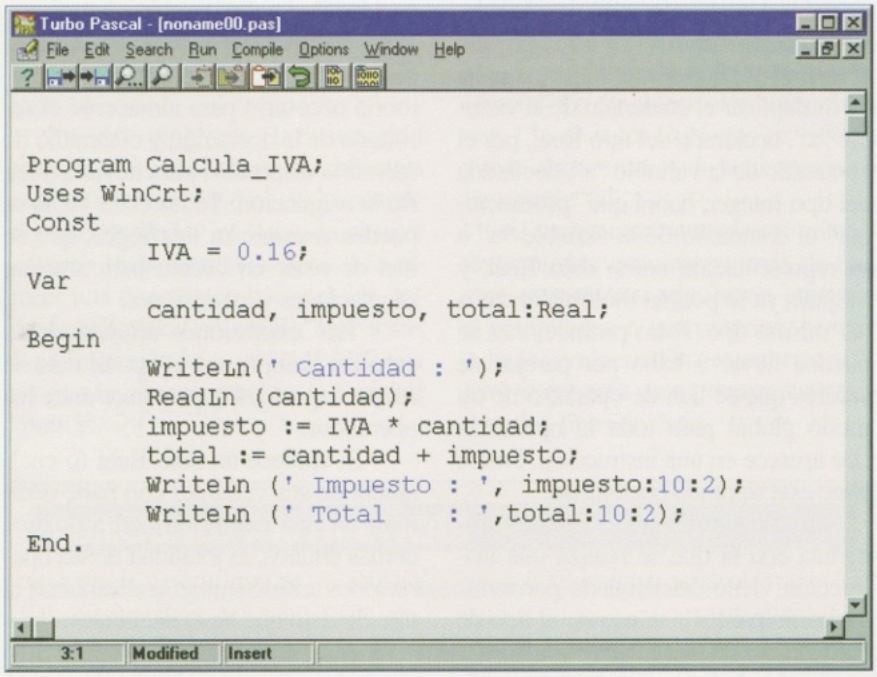


Fig. 14. Programa para el cálculo del incremento debido al IVA.

los resultados para que tengan el aspecto adecuado. La definición de la función permite añadir a las variables el número de espacios que deben ocupar en la salida y, en el caso de los números Real, también determinar el número de cifras decimales que aparecen después del punto. El número de espacios se especifica con un número entero a continuación de la variable y separado por los dos puntos (:). Los espacios que no ocupe el dato aparecerán como huecos en blanco a la izquierda.

En el caso de los datos Real, la segunda cantidad (después de otros dos puntos) indica el número de cifras que han de aparecer a continuación del punto que señala los decimales. Si no se indica esta segunda cifra, la función "WriteLn" asumirá que ha de utilizar la notación científica. Se suele denominar "Especificación de formato de escritura" a la indicación del modo en el que han de aparecer los resultados del programa.

Es muy recomendable trabajar sobre el programa para explorar el funcionamiento de la función "WriteLn", modificando las cantidades que aparecen en la especificación de formato de escritura del programa Calcula_IVA. Abrir una sesión de Turbo Pascal y copiar el programa que puede ver en la figura 14. Compilarlo y ejecutarlo.

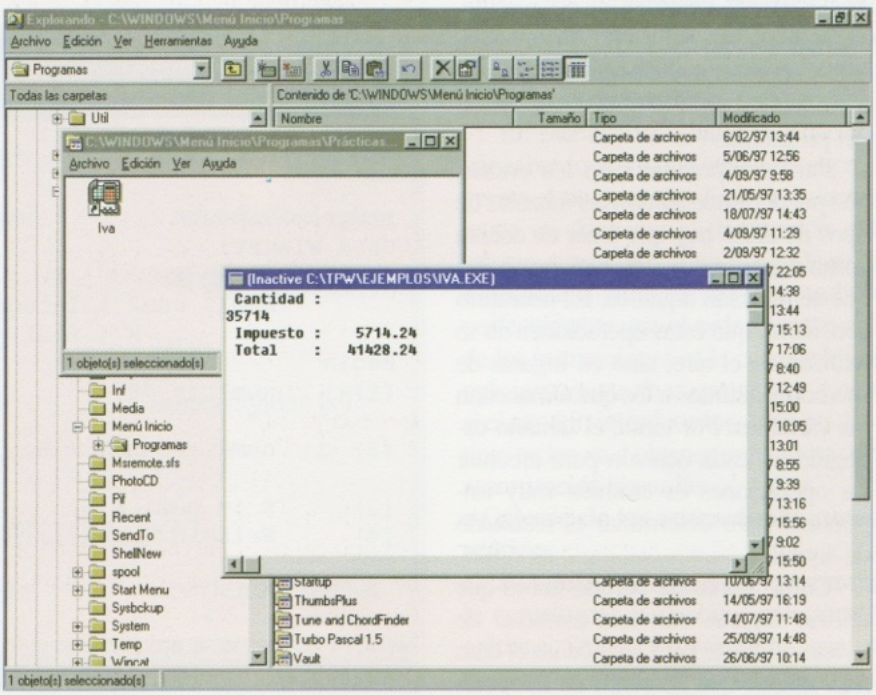
El resultado deberá ser el que aparece en la figura 15: se abre una ventana de Windows en la que se realiza el diálogo y queda constancia de los resultados.

Otra cuestión que puede observarse en la figura 15 es que se ha instalado el programa como una aplicación dentro del grupo de programas de "Practicas de TPW", de modo que se inicia al pinchar el icono elegido en la asignación de iconos desde el menú de propiedades de acceso directo del programa.

Desde este momento ya es posible comenzar a programar aplicaciones que sólo necesiten operaciones aritméticas. Una vez ideada la aplicación, hay que escribir en un papel el diseño del programa. Después se llevará este diseño a líneas de código en las que se utilicen adecuadamente las variables y constantes que se declaren. No hay que dejar de prestar atención al diálogo con el usuario, ni suponer que éste sea el mismo que ha programado la aplicación.

En este estado inicial del aprendizaje de la programación, el mismo compilador de Turbo Pascal para Windows será el tutor, dado que los fallos que se cometan serán previsiblemente de sintaxis y aquél podrá detectarlos.

Fig. 15. Con lo aprendido, ya es posible escribir aplicaciones de cierta utilidad con sólo operaciones aritméticas.



Como también se tratará de operaciones no muy complicadas, una calculadora servirá para comprobar que realmente el programa efectúa los cálculos que se esperan de él.

Como posibles ejercicios se sugieren el cálculo de los minutos y segundos que hay en el número de días que se toma como dato de entrada, la conversión de millas a kilómetros, o la conversión de una cantidad en pesetas a distintas monedas extranjeras. Hay que recordar que siempre deben utilizarse variables en las que los números que se almacenen no excedan su propia capacidad de almacenamiento. La elección más segura son las del tipo Real.

MEZCLA DE TIPOS DE DATOS Y ARITMÉTICA

El diseño del lenguaje Pascal lo hace especialmente adecuado para el aprendizaje. Quizás uno de los aspectos en los que más se puede apreciar esta característica es en la comprobación de tipos de datos tan exhaustiva que realiza el compilador de Pascal antes de dar por bueno un programa.

El compilador avisará cuando se intente asignar a una variable un dato que no pueda almacenar. Pero, de todos modos, no hay que confiarse demasiado puesto que en muchos casos a aquél no le es posible conocer si se va a producir el error durante la ejecución del programa.

Para comprender bien los errores que pueden suceder con las mezclas de tipos de datos hay que tener en cuenta cómo se realizan las operaciones en las que se mezclan aquéllos. Es necesario considerar que estas operaciones no se realizan en el aire, sino en lugares de memoria distintos a los que almacenan las variables. Por tanto, el tamaño escogido en cada ocasión para efectuar las operaciones es también muy importante para determinar el resultado de aquéllas.

Como punto de partida habrá que utilizar lugares con una cantidad de bytes suficiente para almacenar el dato más grande que aparezca en la operación, y el resultado, por precaución, almacenarlo también en un lugar con el mismo tamaño. Por ejemplo, si se ha de multiplicar el contenido de la variable "x", declarada del tipo Real, por el contenido de la variable "a", declarada del tipo Integer, habrá que "promocionar" el contenido de la variable "a" a su representación como dato Real, y después ya se podrán multiplicar datos del mismo tipo. Estas promociones se pueden llevar a cabo por parejas de valores que se han de operar, o de un modo global para toda la operación que aparece en una instrucción. Pascal elige esta segunda posibilidad.

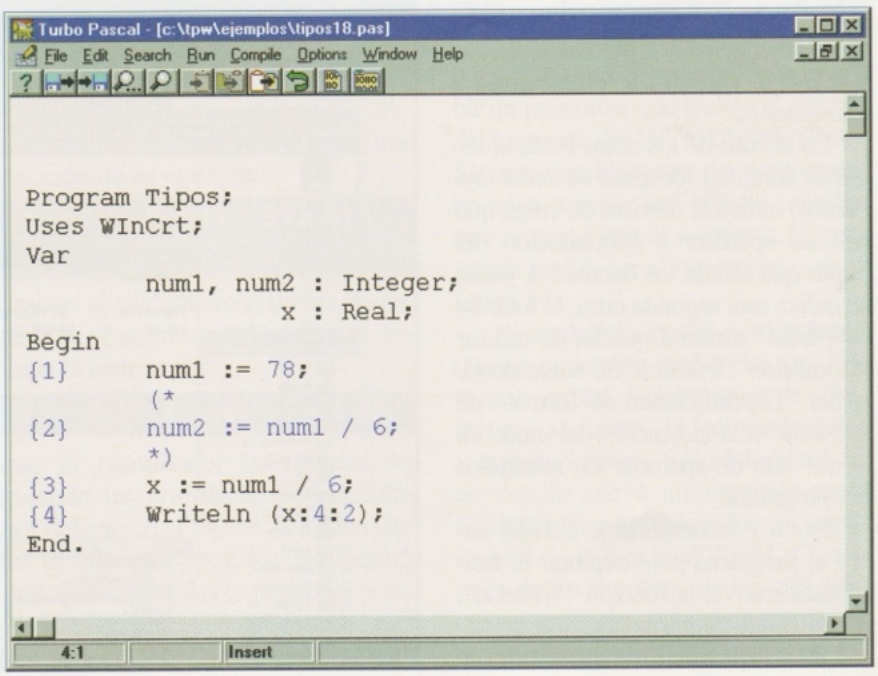
En este programa, el tipo de aritmética con la que se realiza una instrucción viene determinada por todos los participantes y se escoge el tipo de dato de mayor rango para realizar la operación. Así pues, si aparece una sola variable del tipo Real en una expresión aritmética con varios operandos del tipo Integer, toda la operación se realizará con datos Real y, por tanto, intervendrán en ella copias promocionadas a Real de todas las variables Integer que intervengan.

Una vez realizada la operación, el resultado se llevará al lugar de la memoria RAM a que se refiere la variable situada en la parte izquierda de la asignación. Por tanto, también habrá que tener siempre en cuenta las posibles discrepancias entre el tamaño de memoria necesario para almacenar el resultado de la operación y el tamaño de memoria disponible para llevar a buen fin la asignación. Todas estas ideas se pueden resumir en tres reglas que se han de tener en cuenta para analizar asignaciones y expresiones:

- Las expresiones aritméticas se calculan siempre con el tipo de dato de mayor capacidad que aparece entre los operandos.
- Si aparece un dato Real (o cualquiera de sus variantes con parte decimal) en una expresión con todos los demás enteros, la totalidad de las operaciones aritméticas se realizará con el tipo de dato que lleva decimales.
- Aunque no aparezca un Real (o similar) en las operaciones, si aparece el operador de división, todas las operaciones se realizarán con datos del tipo Real.

Esta última regla es bastante razonable, puesto que son escasas las situaciones en las que se pueden dividir números enteros resultando de la operación un número sin decimales.

Fig. 16. Una simple división puede dar lugar a un error si se mezclan variables de distinto tipo.



Para aclarar algunos de estos conceptos se va a utilizar como ejemplo el programa "Tipos" que aparece en la figura 14.

En este programa, la instrucción etiquetada con {2} tiene que colocarse entre comentarios pues, de lo contrario, el compilador avisaría con el error número 26 de mezcla de tipos. Al aparecer una división en la operación, toda la aritmética se realiza con datos Real; el resultado es el Real 13.0 que no puede almacenarse en la variable "num2".

Puede comprobarse que la operación se realiza correctamente si se copia el programa de la figura 16 y se ejecuta. En la instrucción etiquetada con {3} el valor Real que se calcula puede almacenarse adecuadamente en la variable "x".

Al programar operaciones aritméticas pueden encontrarse más de una vez instrucciones que harán protestar al compilador, o que proporcionen resultados erróneos. En general esto ocurrirá muy pocas veces, o nunca si no se mezclan tipos de datos.

FUNCIONES ARITMÉTICAS

Función	Argumento*	Resultado	Operación
Abs(x)	Real o Integer	igual que *	Valor absoluto
Arctan(x)	Real o Integer	Real	Arco tangente
Cos(x)	Real o Integer	Real	Coseno
Exp(x)	Real o Integer	Real	Exponenciación (ex)
Frac(x)	Real o Integer	Real	La parte decimal de x
Int(x)	Real o Integer	Real	La parte entera de x
Ln(x)	Real o Integer	Real	Logaritmo natural
Pred(x)	Integer	Integer	x - 1
Random		Real	Nº aleatorio entre 0 y 1
Random(x)	Integer	Integer	Nº aleatorio entre 0 y x - 1
Round(x)	Real	Integer	x redondeado al entero más cercano
Sin(x)	Real o Integer	Real	Seno
Sqr(x)	Real o Integer	Real	Exponenciación (x^2)
Sqrt(x)	Real o Integer	Real	$\sqrt{x}$
Succ(x)	Integer	Integer	x + 1
Trunc(x)	Real	Integer	x con la parte decimal eliminada

De todos modos, es conveniente estudiar el problema para estar preparados con antelación y así saber interpretar errores inesperados.

ESTUDIO DEL PROGRAMA HIPOTECA

El programa "Hipoteca" tiene la estructura común a muchos otros programas de cálculo de expresiones matemáticas. Aún en uno tan sencillo como éste, es conveniente reconocer claramente sus partes que, como ya se ha dicho, son comunes a muchos programas similares. Reconocer la presencia de estas partes genéricas será muy útil a la hora de escribir nuevos programas. En la figura 17 pueden reconocerse estas partes, que son:

la cabecera
la captura de los datos
El cálculo de las expresiones matemáticas
La presentación de los resultados

- **Cabecera**

Incluye la información que se proporciona al usuario sobre el objetivo del programa. Muchas veces también es conveniente incluir aquí las instrucciones que informen al usuario sobre cómo se debe utilizar la aplicación. En el caso de la cabecera del programa "Hipoteca" que sirve como ejemplo no son necesarias muchas explicaciones.

- **Captura de los datos**

En este apartado hay que reflexionar sobre lo inútil que resultaría el programa si no se antecediera cada lectura de un dato con la descripción del mismo que el programa espera leer. A estas alturas ya se habrá comprendido perfectamente que el orden secuencial de las instrucciones es la característica más importante de un programa, y que si el usuario introduce los datos en distinto orden va a obtener unas cuotas de amortización absurdas.

- **Cálculo de las operaciones matemáticas**

El núcleo principal del programa lo constituyen las operaciones matemáticas. Se puede comprobar que el nombre escogido para las variables

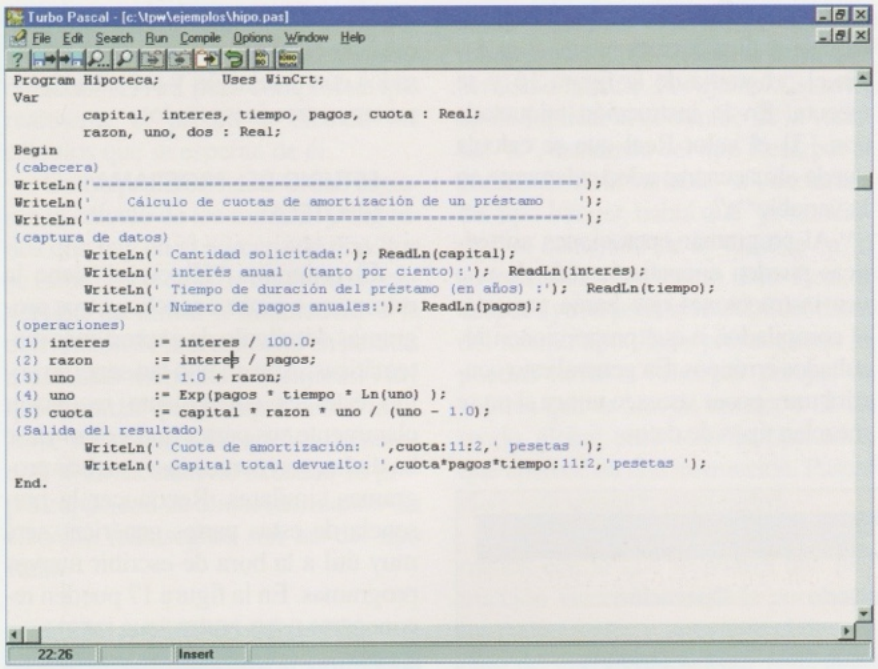


Fig. 17. Programa de cálculo de la cuota de amortización de un préstamo.

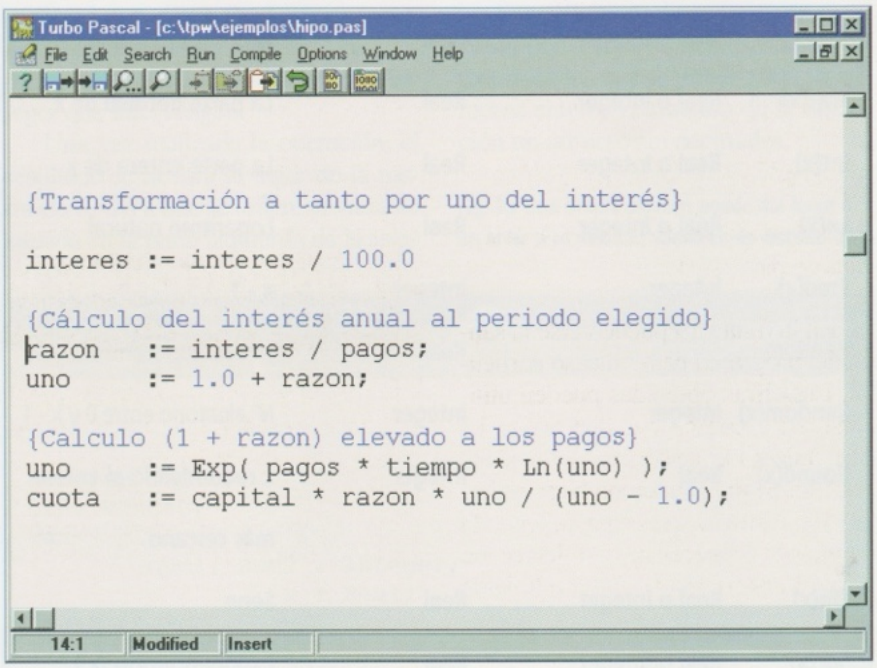
cuyo valor ha de suministrar el usuario, es indicativo de su contenido. Se declaran también las variables "razón", "uno" y "dos" para almacenar resultados de operaciones intermedias, y la variable "cuota" para almacenar el resultado. Todas ellas son, evidentemente, del tipo Real. La fórmula que proporciona la cuota es:

$$\frac{C\,(r/n)(1+(r/n))^{nt}}{(1+(r/n))^{nt}-1}$$

donde **C** es el capital principal, **r** el interés nominal expresado en tanto por uno, **t** el tiempo de duración del préstamo en años, y **n** el número de pagos anuales. Las operaciones que se han de realizar para el cálculo son todas aritméticas, salvo la exponenciación. Turbo Pascal no trae definida ninguna función específica para calcular el resultado de elevar un número cualquiera a otro. La función "Exp" que proporciona eleva el denominado número e (2.7182818...) a un exponente real. Sin embargo, utilizando la función inversa "Ln", que calcula el logaritmo neperiano (con base el número e), se puede obtener la exponenciación deseada. La igualdad matemática utilizada en el programa es:

$$x^y = e^{y \ln x}$$

En la figura 18 pueden verse con más detalle y comentarios las instrucciones de la parte de cálculo del programa:



- En la primera instrucción se reduce el interés de tanto por ciento a tanto por uno.
- En la segunda se calcula la razón entre interés y número de periodos.
- En la tercera se le suma 1 para obtener la base a la que se ha de elevar el número de devoluciones.
- En la cuarta se realiza la exponenciación; hay que comentar el siguiente de-talle: se utiliza para almacenar el resultado del cálculo la misma variable "uno" que se emplea para el argumento de la función que evalúa el logaritmo. Es un ahorro de variables auxiliares que no ensombrece la claridad del programa.
- En la quinta instrucción se realiza el cálculo de la cuota a partir de los resultados intermedios almacenados en la variable "uno".
- **Presentación de los resultados**

El último bloque del programa es la salida de datos.

El formato de escritura de la cuota se elige para que incluya hasta los céntimos y se termina el programa con la cantidad total de dinero que devolvería el banco (sin incluir las comisiones de apertura,...).

Fig. 18. Secuencia de asignaciones y operaciones que permiten calcular la cuota de amortización.

Como ejercicio, escribir el programa "Hipoteca" en la forma que desee cada uno y probar algunos posibles préstamos.

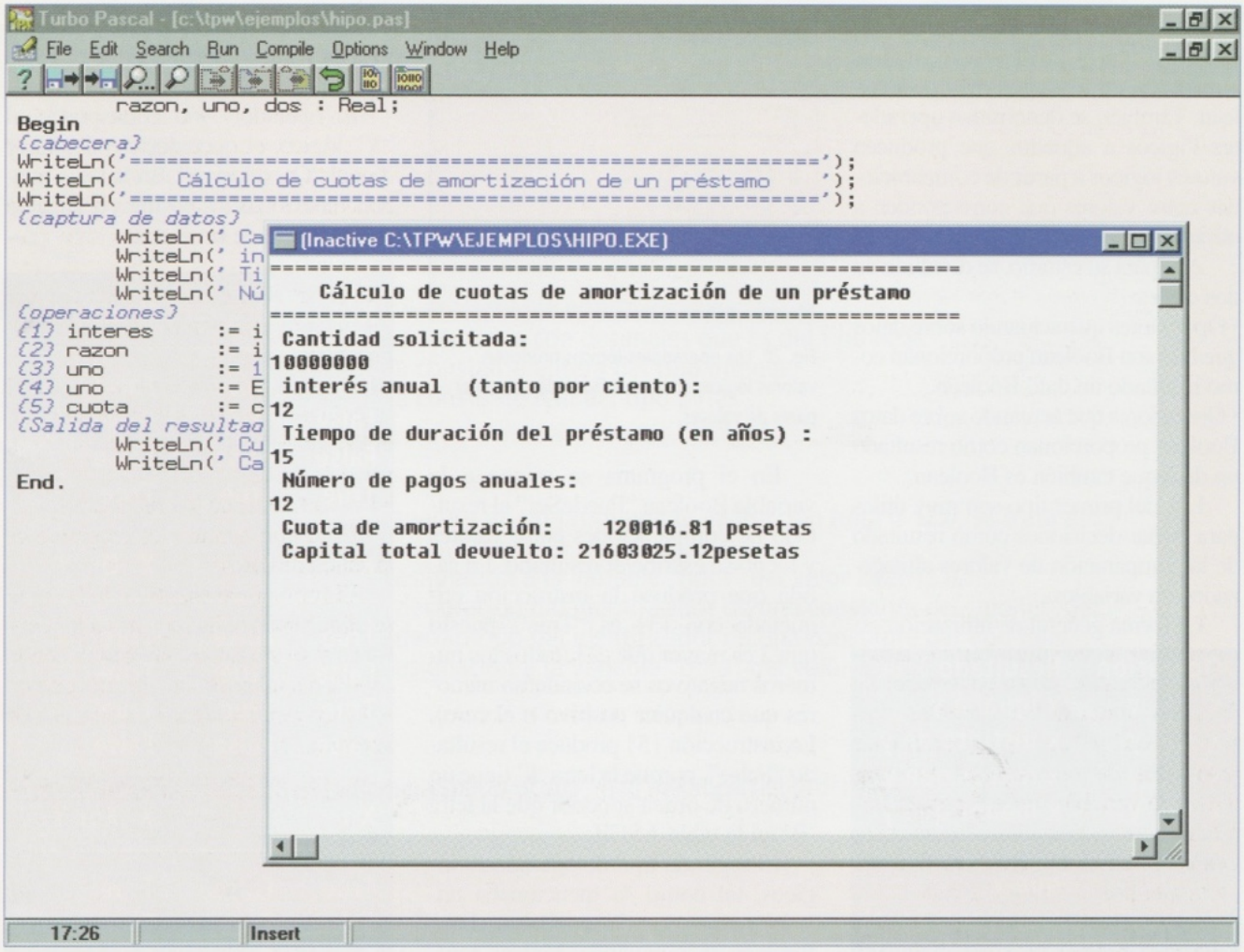


Fig. 19. Salida del programa de cálculo de la cuota de amortización.

En la figura 19 puede verse la salida del programa para un caso particular. Las cifras obtenidas pueden utilizarse para comprobar que no se ha cometido ninguna equivocación al escribirlo. Si todo funciona bien, se habrá demostrado que para obtener la cuota no es necesario ni hacer un viaje al banco, ni utilizar una hoja de cálculo.

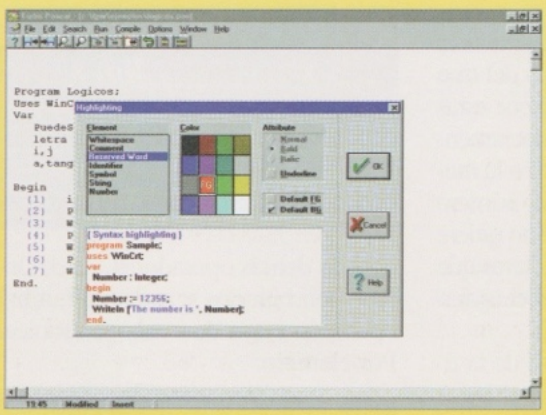
OPERADORES LÓGICOS

Los operadores aritméticos permiten realizar operaciones con los números que se almacenan en los tipos de datos de las familias de los Integer y de los Real.

En esta sección se van a estudiar los operadores lógicos que permiten

MODIFICACIÓN DE LA TIPOGRAFÍA DE LOS PROGRAMAS

El aspecto que presentan los programas no sólo puede modificarse en lo que respecta al tamaño de la tipografía utilizada, sino también en los colores elegidos para resaltar los diferentes componentes de aquél: comentarios, palabras reservadas, indicadores. Para ello, elegir del menú principal la opción Options | Highlighting.



Se encontrará un cuadro de diálogo como el que puede verse en la figura; en él será posible elegir el tipo de componente en la columna de la izquierda y con el ratón asociarle uno de los colores disponibles. En el texto de ejemplo de la parte inferior podrá verse inmediatamente el efecto de la elección.

operar con los valores lógicos de verdadero (True) y falso (False), que se almacenan en los datos del tipo Boolean. También se denominan operadores lógicos a aquéllos que producen valores lógicos a partir de comparaciones entre valores que corresponden a otros tipos de datos.

Así, para su estudio, se dividirán en dos clases:

- Operadores que actuando sobre datos que NO son Boolean proporcionan como resultado un dato Boolean.
- Operadores que actuando sobre datos Boolean proporcionan como resultado un dato que también es Boolean.

Los del primer tipo son muy útiles para tomar decisiones como resultado de la comparación de valores almacenados en variables.

La forma general de utilización es:

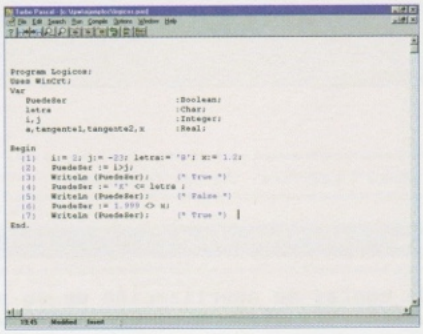
Operación	Resultado True
x = y	x igual a y
x <> y	x distinto de y
x > y	x mayor que y
x á y	x menor que y
x <= y	x menor o igual que y
x >= y	x mayor o igual que y

donde x e y pueden ser números reales, enteros o caracteres, mientras que el resultado de la operación es un dato Boolean (False o True).

La restricción de homogeneidad que exige Pascal para la utilización de estos operadores es que los valores que se comparen han de ser constantes o variables del mismo tipo.

Cuando se utilicen los operadores lógicos para comparar valores del tipo Char, se ha de tener en cuenta que estas comparaciones se realizan directamente sobre la representación que en la memoria tienen los caracteres: el número de orden en la tabla ASCII. Esto quiere decir, por ejemplo, que será cierto que un carácter es mayor que otro si aparece después en la tabla ASCII.

En el programa "Logicos" de la figura 20 pueden verse varios ejemplos de utilización de estos operadores.



```
Program Logicos;
Uses WinCrt;
Var
  PuedeSer                    :Boolean;
  letra                       :Char;
  i,j                         :Integer;
  a,tangente1,tangente2,x     :Real;

Begin
  (1)  i:= 2; j:= -23; letra:= 'B'; x:= 1.2;
  (2)  PuedeSer := (i>j);
  (3)  Writeln (PuedeSer);      (* True *)
  (4)  PuedeSer := 'K' <= letra ;
  (5)  Writeln (PuedeSer);      (* False *)
  (6)  PuedeSer := 1.999 <> x;
  (7)  Writeln (PuedeSer);      (* True *)
End.
```

Fig. 20. Los operadores lógicos producen valores lógicos como resultado de comparar pares de valores.

En el programa se asigna a la variable Boolean "PuedeSer" el resultado de comparaciones entre valores y luego se escribe el resultado. La salida que produce la instrucción etiquetada con {3} es "True", puesto que 2 es mayor que -23 (todos los números negativos se consideran menores que cualquier positivo o el cero). La instrucción {5} produce el resultado "False", porque la letra 'k' tiene un número de orden superior que la letra 'B' en la tabla ASCII.

El segundo tipo de operadores lógicos, tal como se mencionaba anteriormente, permite combinar datos Boolean en expresiones lógicas complejas.

Uno de ellos, Not, es monario y se utiliza según la forma general:

```
Not operando
```

donde "operando" es un indicador que se refiere a un dato del tipo Boolean y el resultado es la negación del valor del operando, tal como se especifica en la tabla:

x	Not x
True	False
False	True

Los demás operadores, And, Or, y Xor, son binarios y se han de utilizar operando entre dos valores Boolean. Por ejemplo:

```
x And y
```

y el resultado de la operación será cierto si es verdad la expresión lógica que representan.

El operador And corresponde al "Y" lógico, el operador Or al "O", y Xor al "O exclusivo". En la tabla pueden verse todas los posibles resultados dependiendo de los casos.

X	Y	X And Y
True	False	False
False	True	False
False	False	False
True	True	True

X	Y	X Or Y
True	False	True
False	True	True
True	True	True
False	False	False

X	Y	X Xor Y
True	False	True
False	True	True
True	True	False
False	False	False

RESUMEN

Las variables de un programa han de declararse y clasificarse según los tipos de datos básicos permitidos por el lenguaje de programación. De esta forma, según el tipo de dato al que pertenezcan, intervendrán de un modo distinto en las operaciones que se programen utilizando operadores aritméticos o lógicos. Como norma general, se ha de tratar de no mezclar tipos de datos en las expresiones aritméticas y, en caso contrario, habrá que tener siempre presente las características del resultado al que dicha mezcla da lugar según las operaciones que intervengan en la expresión.

CUESTIONARIO AUTOEVALUATIVO

1 - Cuando en un programa Pascal aparece 536.0, se trata de:

- ❑ A - Un valor variable entero.
- ❑ B - Una constante con decimales que es del tipo Real.
- ❑ C - Una constante entera del tipo Integer.
- ❑ D - Una constante real del tipo Byte.

2 - Si aparece en un programa la instrucción hora = 12.0, se trata de:

- ❑ A - La asignación a la variable "hora" del valor Real 12.0.
- ❑ B - La declaración del identificador de una constante del tipo Integer.
- ❑ C - La declaración del identificador de una constante del tipo Real.
- ❑ D - La asignación a la variable "hora" del valor Integer 12.

3 - ¿Qué espacio ocupa en la memoria del ordenador una variable del tipo Char?

- ❑ A - 1 bit.
- ❑ B - 1 byte.
- ❑ C - 2 bits.
- ❑ D - 2 bytes.

4 - La constante 33767.0 puede almacenarse en una variable del tipo:

- ❑ A - Real.
- ❑ B - Integer.
- ❑ C - LongInt.
- ❑ D - Word.

5 - El resultado de la operación 12 / 2 puede almacenarse en una variable del tipo:

- ❑ A - Real.
- ❑ B - Integer.
- ❑ C - LongInt.
- ❑ D - Word.

6 - El resultado que devuelve la llamada a la función Sqr (9) es:

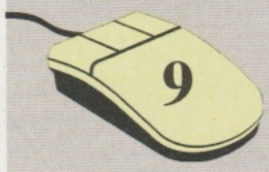
- ❑ A - 3.
- ❑ B - 99.
- ❑ C - 2.
- ❑ D - 81.0.

7 - El resultado de la operación 26 Div 5 es:

- ❑ A - 5.2.
- ❑ B - 5.11.
- ❑ C - 5.
- ❑ D - 1.

8 - Cuando "x" almacena el valor 'h' ¿qué ocurre con la expresión Not (('a' >= x) Or ('z' > x))?

- ❑ A - Da como resultado True.
- ❑ B - Es incorrecta sintácticamente.
- ❑ C - Da lugar a un error en ejecución.
- ❑ D - Da como resultado False.



9 - Cuando la variable "x" se ha declarado del tipo String, ¿qué ocurre con la asignación x := 'True';?

- ❑ A - Es incorrecta y el error lo detectará el compilador.
- ❑ B - Es correcta.
- ❑ C - Es incorrecta y el error se detectará en ejecución.
- ❑ D - Es correcta o no dependiendo del valor almacenado anteriormente en la variable "x".

10-Cuando la variable "x" se ha declarado del tipo Integer, ¿qué ocurre con la asignación x := 224 * 210 ;?

- ❑ A - Es incorrecta y el error lo detectará el compilador.
- ❑ B - Es correcta.
- ❑ C - Es incorrecta y el error se detectará en ejecución.
- ❑ D - Es correcta o no dependiendo del valor almacenado anteriormente en la variable "x".

Respuestas: 1 B - 2 C - 3 B - 4 A - 5 A - 6 D - 7 C - 8 D - 9 B - 10 A